

PLP - Primer Parcial - 2^{do} cuatrimestre de 2022

Este examen se aprueba obteniendo al menos dos ejercicios bien menos (B-) y uno regular (R). Las notas para cada ejercicio son: -, I, R, B-, B. Entregar cada ejercicio en hojas separadas. Poner nombre, apellido, número de orden y cantidad de hojas en la primera hoja, y numerar las hojas. Se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras.

El orden de los ejercicios es arbitrario. Recomendamos leer el parcial completo antes de empezar a resolver.

Ejercicio 1 - Programación funcional

Aclaración: en este ejercicio no está permitido utilizar recursión explícita.

Las matrices infinitas pueden ser representadas como funciones:

```
type MatrizInfinita a = Int -> Int -> a
```

donde el primer argumento corresponde a la fila, el segundo a la columna y el resultado al valor contenido en la celda correspondiente.

Por ejemplo, las siguientes definiciones:

```
identidad = \i j->if i==j then 1 else 0
```

```
cantor = \x y->(x+y)*(x+y+1) `div` 2 + y
```

```
pares = \x y->(x,y)
```

corresponden a las matrices:

1	0	0	...
0	1	0	...
0	0	1	...
⋮	⋮	⋮	⋱

identidad

0	2	5	...
1	4	8	...
3	7	12	...
⋮	⋮	⋮	⋱

cantor

(0,0)	(0,1)	(0,2)	...
(1,0)	(1,1)	(1,2)	...
(2,0)	(2,1)	(2,2)	...
⋮	⋮	⋮	⋱

pares

Definir las siguientes funciones:

- `comparar::Eq a => MatrizInfinita a -> MatrizInfinita a -> MatrizInfinita Bool`, que compara dos matrices infinitas posición por posición, y devuelve una tercera cuyos valores son `True` en las posiciones que son iguales en ambas matrices, y `False` en las que son diferentes.
- `recortar::Int -> Int -> MatrizInfinita a -> [[a]]` que, dados dos enteros f y c y una matriz m , devuelve la submatriz finita de m (representada como lista de filas) con las primeras f filas y c columnas. **Pista:** tener en cuenta que es posible anidar listas por comprensión.
Por ejemplo: `recortar 3 5 cantor` devuelve `[[0,2,5,9,14],[1,4,8,13,19],[3,7,12,18,25]]`.
- `hayRepetidoHasta::Int -> Int -> MatrizInfinita a -> Bool`, que indica si hay algún elemento repetido en las primeras f filas y c columnas de la matriz.

Ejercicio 2 - Cálculo Lambda Considerar el Cálculo Lambda tipado extendido con listas. Se desea extender el cálculo para poder construir listas a partir de un primer elemento, una función generadora y una condición de corte.

El conjunto de tipos no se modifica. El conjunto de términos se extiende de la siguiente manera:

$$M ::= \dots \mid \text{from } M \text{ until}_x M \text{ by } M$$

Una expresión de la forma $\text{from } M_1 \text{ until}_x M_2 \text{ by } M_3$, donde M_1 es un elemento cualquiera, M_2 es una expresión booleana que puede tener libre la variable x , y M_3 una función que se aplicará a cada elemento para generar el siguiente, reducirá a una lista cuyo primer elemento - de haberlo - es el valor de M_1 , y cada elemento subsiguiente se obtiene aplicando M_3 al elemento anterior hasta que se satisfaga la condición M_2 tomando como x al elemento actual. Si M_2 es verdadera al tomar M_1 como x , entonces la lista resultante será vacía.

Por ejemplo:

$\text{from } 0 \text{ until}_x \text{if isZero}(x) \text{ then False else Not isZero}(\text{Pred}(x)) \text{ by } \lambda n: \text{Nat.Succ}(n) \rightarrow 0 :: []_{\text{Nat}}$

$\text{from False until}_y y \text{ by } \lambda b: \text{Bool.Not } b \rightarrow \text{False} :: []_{\text{Bool}}$

$\text{from False until}_z \text{True by } \lambda b: \text{Bool.} b \rightarrow []_{\text{Bool}}$

- Dar la(s) regla(s) de tipado para soportar los nuevos términos.
- Describir el conjunto de valores y dar la(s) regla(s) de reducción en un paso para los nuevos términos.
- Mostrar paso a paso cómo reduce la expresión:

$\text{from } 0 \text{ until}_x \text{if isZero}(x) \text{ then False else True by } \lambda n: \text{Nat.Succ}(n)$

- Definir como macro la función sucesiónAritmética, que dados tres naturales - el primer elemento, el incremento y la cota superior - genera la sucesión aritmética correspondiente. Por ejemplo:

sucesiónAritmética $1 \ 4 \ 10 \rightarrow 1 :: 5 :: 9 :: []_{\text{Nat}}$

sucesiónAritmética $2 \ 3 \ 5 \rightarrow 2 :: 5 :: []_{\text{Nat}}$

Suponer definidos los operadores $<$, $>$, $=$, \leq , \geq y $+$ para el tipo Nat.

Ejercicio 3 - Subtipado

Considerar el Cálculo Lambda tipado extendido con listas. Se desea extender el cálculo para soportar procesadores de listas. Es decir, expresiones que realicen algún tipo de cómputo sobre una lista para devolver un valor.

Se extenderán los tipos y términos de la siguiente manera:

$$\sigma ::= \dots \mid \text{Proc}_{\sigma,\sigma} \quad M ::= \dots \mid \text{procesar}_{\sigma} [] \rightsquigarrow M; h :: t, r \rightsquigarrow M \mid M \langle\langle M \rangle\rangle$$

$\text{Proc}_{\sigma,\tau}$ es el tipo de los procesadores que esperan listas de tipo $[\sigma]$ y devuelven resultados de tipo τ .

Las reglas de tipado son las siguientes:

$$\frac{\Gamma \triangleright M : \tau \quad \Gamma, h : \sigma, t : [\sigma], r : \tau \triangleright N : \tau}{\Gamma \triangleright \text{procesar}_{\sigma} [] \rightsquigarrow M; h :: t, r \rightsquigarrow N : \text{Proc}_{\sigma,\tau}} \text{(T-PROC)} \quad \frac{\Gamma \triangleright M : \text{Proc}_{\sigma,\tau} \quad \Gamma \triangleright N : [\sigma]}{\Gamma \triangleright M \langle\langle N \rangle\rangle : \tau} \text{(T-APPPROC)}$$

- Dar la(s) regla(s) de subtipado para esta extensión y justificar en términos del principio de sustitutividad.
- Utilizando las reglas de tipado y subtipado, derivar el siguiente juicio de tipado:

$$\{xs : [\text{Int}]\} \triangleright (\lambda p : \text{Proc}_{\text{Int},\text{Int}.p} \langle\langle xs \rangle\rangle) (\text{procesar}_{\text{Float}} [] \rightsquigarrow 0; h :: t, r \rightsquigarrow \text{Succ}(r)) : \text{Float}$$